Constraint-Based Off-Nominal Behavior Modeling for Europa Clipper

Anthony Zheng, Bradley J. Clement, David K. Legg,
Michel D. Ingham, Kelli J. McCoy, Chester J. Everline
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
zhengant@umich.edu, leggd@uci.edu, *first.[middle.]last*@jpl.caltech.edu
818-393-4729

Abstract— **The risk analysis for the Europa Clipper mission evaluates the probability of mission failure based on the failure rates of individual components and dependencies among them. The probabilities are calculated by integrating over the intervals of time within which a fault occurs, accounting for an infinite number of cases. The response of the spacecraft to different faults can result in different schedules of activities, changing the intervals of integration. Europa currently uses models of spacecraft systems and components to simulate individual flight scenarios. The goal is to develop a framework for integrating, automating, and improving this modeling process.**

We describe an approach to generating the schedules for the different fault cases and determining the intervals for faults. It is not enough to just simulate individual cases because we are working with continuous variables that generate an infinite number of possible futures. Instead, we determine time windows within which certain faults can occur and use these time windows as bounds for integration. We found that determining these time windows is a constraint optimization problem.

In order to represent these problems, we employ a language based on ontologies of behavior and scenarios. The language enables us to specify constraints in a simple, declarative syntax. A constraint-based analysis engine uses the declarative specification to identify bounds on system parameters and fill in details of behavior.

For example, we created a detailed model of power generation, power use, and the corresponding effects on the battery in order to determine when an undervoltage fault can occur. An undervoltage during a trajectory correction maneuver requires that thrusting be interrupted for just enough time to recharge the battery such that the maneuver can be completed within battery limits. This behavior is generated based on the model to minimize the interruption time.

For certain scenarios the constraint optimization problems were simple enough to be solved by hand, but the framework made the process substantially faster. It also produced solutions to other problems that we could not solve by hand or with existing tools and allowed us to generate and run many scenarios at once. The scenario language and engine greatly simplified the process of identifying time bounds and separating cases.

978-1-5386-6854-2/19/$31.00 ©2019 IEEE

TABLE OF CONTENTS

1. INTRODUCTION

The Probabilistic Risk Analysis (PRA) for the Europa Clipper mission evaluates the probability of mission failure based on the failure rates of individual components and their relationships. PRAs for space missions often assess risk with abstract mathematical models of faults and closed-form formulas, with which probabilities can be calculated [1][4][5]. Such formulas can often be computed very quickly, which allows for easy validation of results and identification of problems in the model, but developing these formulas becomes more challenging as increasing detail about faults and responses is incorporated. One approach to handle this complexity is to model the functional relationships among components and subsequently generate the closed-form formula from that model [12], which we refer to as Model-Based PRA (MBPRA). This approach calculates the probability that the spacecraft is reliable as a function of time, but relies on a nominal schedule and assumes that swapping to redundant components takes insignificant time and that recovery can be assessed based on relatively simple deviations from the nominal schedule (see [9] for an example of this approach).

Analysis of the Europa Clipper Jupiter Orbit Insertion (JOI) sub-phase, in which the spacecraft executes a maneuver to enter Jupiter's orbit, is complicated by the fact that some

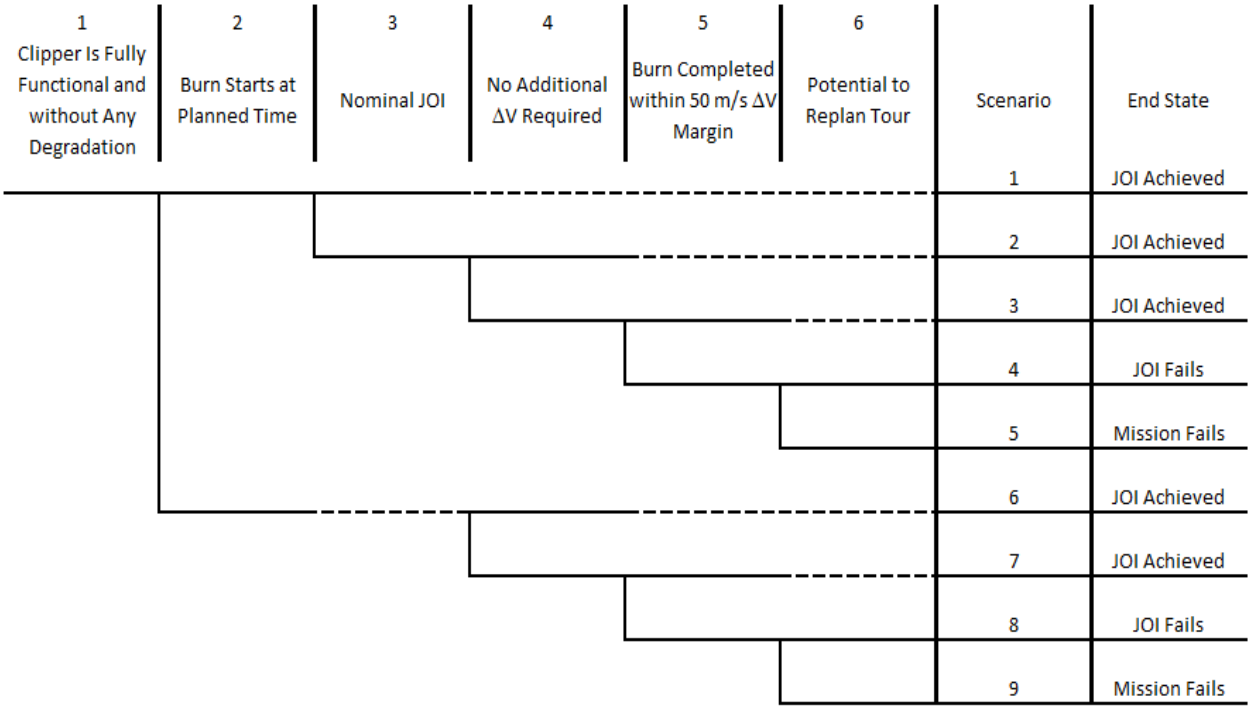


Figure 1: Event tree for JOI. Scenarios 1-4 and 6-8 represent scenarios where mission success is still possible. Scenario 1 is nominal and the rest are off-nominal. The probability of mission failure is calculated as (1-P) where P is the total probability of one of Scenarios 1-4 or 6-8 occurring. Modeling Scenarios 6-8 is analogous to Scenarios 2-4, just with a different start time.

faults cause significant changes to the JOI schedule. During JOI, thruster failures can cause delays which increase the amount of Δv the spacecraft must achieve to complete JOI, extending the burn duration. It is possible that such a delay would exhaust the battery, requiring the spacecraft to stop burning long enough to recharge, before continuing the maneuver. Such major changes to the nominal schedule are not readily captured by a single closed-form formula, nor by traditional MBPRA techniques. Furthermore, exhaustive simulation is impossible, since thruster faults can occur at any time during the burn(s), effectively generating an infinite number of possible timelines, and non-exhaustive simulation of multi-component systems is difficult to do accurately [11].

We capture this behavior by grouping possible futures such that faults occur in well-defined time windows, over which the probability of thruster faults can be integrated. Determining these time windows can be framed as a constraint optimization problem, which we model using a concise, declarative constraint language.

Furthermore, we describe a way to integrate this framework with existing Model-Based Systems Engineering (MBSE) tooling, including a web-based interface for diagrammatic modeling and a textual modeling language that share a common behavior-based ontology [8]. This moves towards the Single-Source-of-Truth paradigm, intended to reduce inconsistencies between models, and to clarify communication between parties working on different aspects of the mission [2]. This integration allows for several different analyses, including those on planetary protection [1], power management [10], and science sensitivity [9], to share a common model of the mission and remain in agreement with design requirements as design details evolve.

2. Risk Modeling Challenge

Since MBPRA relies on the output of a simulation of nominal spacecraft behavior to determine when components are active, it cannot directly represent fault recoveries that change the behavior of the spacecraft over a long period of time. For Europa Clipper JOI, faults in the propulsion subsystem could result in degraded behavior, and, if the total thrust goal is not completed within a particular time frame, then the mission could fail. Furthermore, multiple successive thruster faults could occur, each increasing the degradation of spacecraft behavior, possibly by different levels.

The timing and succession of faults do not fit the MBPRA model in two ways. First, the spacecraft's dependence on the timing and level of degradation cannot be captured. Second, a longer duration of thrust could result in depletion of the battery, triggering an interruption in thrust to recharge. This alternative course of events violates MBPRA's assumption of a nominal schedule.

Thus, the propulsion subsystem is treated as a special case. To incorporate its behavior into the calculation, reliability is

integrated over all possible sequences of faults. For example, suppose that there are three successive faults, each resulting in further degraded thrust capability. If the timing of the second and third faults depend on the times of the first and second, respectively, then the reliability is calculated with three nested integrals. A deadline, T, for the thrust to complete can bound the times of the faults. Now if the thrust goal is achieved (requiring 6.5 hours) before the first fault occurs then the faults do not matter. So, the upper bound of the first integral is 6.5 hours. If there is a fault at time, x, within 6.5 hours, the thrust becomes degraded and drawn out over a longer period. So, the completion time if no second fault occurs is a function, $S_1(6.5 - x)$, the remaining time to reach 6.5 after the first fault multiplied by a slowdown factor of S_1. This is the upper bound of the integral for the second fault. Similarly, if there is a second fault before the completion, and the slowdown factor increases to S_2, then the completion time is $S_2\left(6.5 - x - \frac{1}{S_1}y\right)$, the upper bound of the third integral. Thus, the nested integral for calculating reliability over the period of the thrust based on some function, $r(x, y, z, T)$, is

$$\int_0^{6.5} \int_x^{S_1(6.5-x)} \int_y^{S_2\left(6.5-x-\frac{1}{S_1}y\right)} r(x, y, z, T)\, dz\, dy\, dx \quad (1)$$

The next section shows that in our case, the calculation of these bounds for different scenarios is actually more complicated.

The integration over fault times is the strategy for capturing the more complicated fault behavior in closed-form, but $r(x, y, z, T)$ is complex, and it is unclear how to obtain its closed-form. $r(x, y, z, T)$ is based on the MBPRA model, and a reliability formula is generated based on a nominal simulation. However, that simulation depends on x, y, and z, and can vary dramaticallyif the battery is depleted. Thus, the reliability formula output of MBPRA must account for an infinite set of these simulations, which it cannot. And even if it could, the formula may be very large to capture different combinations of behaviors. In the actual implementation, an abstraction of the behavior is used to avoid this problem. But, it is an open problem for which we do not have a general strategy.

3. JOI THRUSTER FAULT RECOVERY MODEL

In this section, we describe how we incorporate the possibility of thruster faults into a model of JOI completion. This model is used later to compute bounds for the integrals described in Section 2. Note that the model is hypothetical and is unlikely to reflect the actual fault behavior of Europa Clipper.

There are nine scenarios explored for JOI as shown in Figure 1. We focused on modeling the first four scenarios since successfully modeling these gives us the foundational capabilities to model the rest of the scenarios as well. Scenarios 6 – 9 involve a delay in the burn start that we do not include in this model but discuss in Section 5.

Scenario 1 represents a nominal JOI, defined as completing the 900 m/s burn between 6.5 and 7.5 hours. In Scenario 2, the spacecraft fails to complete JOI within 7.5 hours due to some delay but completes it in less than 13.4 hours. If the delay takes the spacecraft past 13.4 hours, the spacecraft will need to burn for some additional Δv to complete JOI. The amount of additional Δv necessary is a function of the amount of delay and the amount of the burn already completed and is modeled as a piecewise linear function. If the amount of additional Δv necessary is less than 50 m/s, then the spacecraft is in Scenario 3. Otherwise, the spacecraft is in Scenario 4. These scenario definitions are summarized in Table 1.

With no thruster losses, the spacecraft would complete the entire burn on 8 thrusters. When the spacecraft experiences thruster faults, it does so in pairs to maintain thrust symmetry. We identify six additional cases to model: starting nominally, with 8 thrusters, then experiencing a fault in one pair (8 to 6); Starting with eight, and experiencing faults in two pairs sequentially (8 to 6 to 4); Starting sub-nominally, with 6, but experiencing no faults (6); Starting with 6 and experiencing one fault (6 to 4); and starting with only 4, but experiencing no faults (4). This gives us a total of seven cases for each of the four scenarios.

Table 1: Summary of constraints for Scenarios 1-4

	Scenario 1	Scenario 2	Scenario 3	Scenario 4
JOI Completion Time (hours)	$6.5 \leq t \leq 7.5$	$7.5 \leq t \leq 13.4$	$t > 13.4$	$t > 13.4$
Additional Δv (m/s)	$\Delta v = 0$	$\Delta v = 0$	$0 < \Delta v < 50$	$\Delta v > 50$

Let $\Delta v_{nominal}$ be a constant equal to the amount of Δv (change in velocity) necessary to complete a nominal burn. Define t_k as the amount of time spent burning on k thrusters. Define $t_k^{(1)}$ as the amount of time spent burning using k thrusters during the first burn (before any delay). Define t_{delay} as the amount of time between the two burns and C as the percentage of the nominal burn, ignoring any additional Δv requirement, completed before the delay. Define Δv_{add} as the amount of additional Δv added to correct for the delay. We evaluate burn completion based on thruster-hours, where 1 thruster-hour is defined as the progress on a burn made by burning on one thruster for one hour. A complete nominal burn, not including any additional burn necessary for the additional Δv, is defined as $\tau_{nominal}$ thruster-hours, where $\tau_{nominal}$ is constant. Defining τ_{add} to be the additional thruster hours required because of the additional Δv, we have the following constraints:

$$\tau_{add} = \frac{\Delta v_{add} * \tau_{nominal}}{\Delta v_{nominal}} \quad (2)$$

$$8t_8 + 6t_6 + 4t_4 = \tau_{nominal} + \tau_{add} \quad (3)$$

Similarly, we also have that

$$C = \frac{t_8^{(1)} + t_6^{(1)} + t_4^{(1)}}{\tau_{nominal}} \quad (4)$$

For simplicity, we assume that the rate at which the battery discharges and recharges is constant throughout this scenario, regardless of the number of thrusters burning or the angle of the spacecraft relative to the sun. A much larger model (not discussed here) is used to determine more precisely how power and battery state change over time. For this more abstract model, this means that the time at which the spacecraft undervolts is constant, $T_{undervolt}$. Using $T_{undervolt}$, we can get expressions for the $t_k^{(1)}$:

$$t_8^{(1)} = \min(t_8, T_{undervolt}) \quad (5)$$

$$t_6^{(1)} = \min\left(t_6, T_{undervolt} - t_8^{(1)}\right) \quad (6)$$

$$t_4^{(1)} = \min\left(t_4, T_{undervolt} - t_8^{(1)} - t_6^{(1)}\right) \quad (7)$$

The delay should be just long enough so that the spacecraft can complete the second burn. Because the discharge and recharge rates are constant, the length of delay is directly proportional to the length of burn:

$$t_{delay} = \mathrm{R}\left(T_{undervolt} - (t_8 + t_6 + t_4)\right) \quad (8)$$

where R is some constant.

We model Δv_{add} as follows:

$$t_{delay} = \mathrm{R}\left(T_{undervolt} - (t_8 + t_6 + t_4)\right) \quad (8)$$

```
// objective
var min_t8 : Real = minimize("t8")

// constants: assume these are given some value
var nominalDeltaV : Real
var tau_nominal : Real
var C : Real
var T_undervolt : Real
var R : Real
var D : Real
// declarations for m1...m4 and T1...T4 omitted
for brevity

// declare non-constant variables
var t8 : Real
var t6 : Real
var t4 : Real
var t8_1 : Real
var t6_1 : Real
var t4_1 : Real
var t_delay : Real
var deltaVAdded: Real
var tau_add : Real
var m : Real

// keep values positive
req t8 >= 0
req t6 >= 0
req t4 >= 0

// model the constraints
req tau_add = deltaVAdded *
              tau_nominal / nominalDeltaV
req 8*t8 + 6*t6 + 4*t4 = tau_nominal + tau_add
req C = (t8_1 + t6_1 + t4_1) / tau_nominal
req t8_1 = min(ta, T_undervolt)
req t6_1 = min(tb, T_undervolt, t8_1)
req t4_1 = min(tc, T_undervolt, t8_1, t4_1)
req t_delay = R*(T_undervolt - (t8 + t6 + t4))
req deltaVAdded = m(1 - C)
req deltaVAdded > 0 && deltaVAdded < D

// interpolation
req m =
  if t_delay >= T1 && t_delay < T2
  then m1 + (t_delay - T1)*(m2 - m1) /
            (T2 - T1)
  else
    if t_delay >= T2 && t_delay < T3
    then m2 + (t_delay - T2)*(m3 - m2) /
              (T3 - T2)
    else m3 + (t_delay - T3)*(m4 - m3) /
              (T4 - T3)
```

Figure 2: An example model of our optimization problem written in K.

where m is based on t_{delay}: we have constants $m_1, \ldots, m_4$ and $T_1, \ldots, T_4$ such that $T_1 \leq t_{delay} \leq T_4$, $m_i \leq m_j$ and $T_i < T_j$ for $i, j \in \{1, \ldots, 4\}$, $i < j$, and that m is linearly interpolated between the m_k for $t_{delay} \notin \{1, \ldots 4\}$. The objective in this example is find the minimize the amount of time spent burning on eight thrusters such that $0 < \Delta v_{add} < D$, where D is some constant.

The elements of the model described so far are common across all the scenarios; specific models for each can be created by adding the additional constraints from Table 1 to restrict the model to a particular scenario.

4. SOLVING FRAMEWORK

To generate the closed-form formula for the reliability of the spacecraft during JOI, we separate the JOI sub-phase into mutually exclusive scenarios with specific sequences of faults. We have expressions for risk in terms of the amount of time spent on a certain number of thrusters, which we integrate over the time bounds within which a certain scenario can occur [3][4]. The limits on these integrals are the maximum and minimum amounts of time the spacecraft can spend on a particular number of thrusters while still satisfying the constraints of a given scenario. This means that we can frame this problem as an optimization problem where we iteratively maximize and minimize the amount of time spent on 8 thrusters, 6 thrusters, and 4 thrusters for each scenario and thruster case. In total, we have six objectives, seven thruster cases, and four scenarios, for a total of 168 individual optimization problems to solve.

We used the K language paired with the Behavioral Analysis Engine (BAE) to model and solve the optimization problems[1]. The K language is a declarative language for describing variables and constraints on the values of those variables (Havelund, et al. 2016). The K language is linked into BAE, a constraint-based solving engine that can take these constraints and find a point solution that satisfies the constraints, if such a solution exists, and optimize for minimum or maximum value of a variable. It does not guarantee completeness and optimality in general, but it does for our particular set of problems.

To iteratively run the optimization problems, we wrapped the K and BAE solver system with code that generates the K models on the fly based on an existing SysML model, adding the corresponding constraints to a set of base constraints that describe the physics independent of the specific scenario.

Additionally, we have tooling to generate descriptions in K of models described using an ontology of behavior shared by other modeling tools, including BeCoS, an online diagrammatic modeling tool [8]. This shared ontology makes mechanical translation between diagrammatic and textual representations efficient and accurate. Furthermore, the linkage between this ontology and the K/BAE analysis framework helps reduce miscommunication between design engineers and risk analysis team members.

Table 2: Bounds on the amount of time spent burning on a certain number of thrusters for Scenario 2.

Case	Constraints	Δt_8	Δt_6	Δt_4
8 thrusters throughout	$\Delta t_8 > 0 \Delta t_6 = 0 \Delta t_4 = 0$	No solution		
Switch from 8 to 6 thrusters	$\Delta t_8 > 0 \Delta t_6 > 0 \Delta t_4 = 0$	$0.62 < \Delta t_8 < 3.5$	$4.0 < \Delta t_6 < 7.84$	0.0
Switch from 8 to 6 to 4 thrusters	$\Delta t_8 > 0 \Delta t_6 > 0 \Delta t_4 > 0$	$0.62 < \Delta t_8 < 5.5$	$0.0 < \Delta t_6 < 7.84$	$0.0 < \Delta t_4 < 3.92$
Switch from 8 to 4 thrusters	$\Delta t_8 > 0 \Delta t_6 = 0 \Delta t_4 > 0$	$4.54 < \Delta t_8 < 5.5$	0.0	$2.0 < \Delta t_4 < 3.92$
6 thrusters throughout	$\Delta t_8 = 0 \Delta t_6 > 0 \Delta t_4 = 0$	No solution		
Switch from 6 to 4 thrusters	$\Delta t_8 = 0 \Delta t_6 > 0 \Delta t_4 > 0$	No solution		
4 thrusters throughout	$\Delta t_8 = 0 \Delta t_6 = 0 \Delta t_4 > 0$	No solution		

[1] K and BAE are available at https://github.com/Open-MBEE/kservices.

5. RESULTS AND ANALYSIS

Table 2 shows the time bounds calculated by BAE for the K models for Scenario 2. "No solution" for a certain case means that there are no times that the thrusters can fault in the way the case describes and still be within the scenario. For this example, this means that to be in Scenario 2, the spacecraft must spend at least some time (0.62 hours) burning on 8 thrusters, but not too much (5.5 hours maximum).

For Scenarios 1 and 2, the optimization problems are simple enough that we are able to validate those results against hand calculations (see Appendix A).

We found that using the K/BAE framework provided a number of important advantages compared to other approaches. First, because of K's simple and expressive syntax, our models were easy to write and especially easy to generate. At some point, we attempted to feed the optimization problems into established linear or quadratic programming solvers, but they all resulted in large, difficult-to-read, and unintuitive models. The K language, on the other hand, supports syntax like "if then else" for conditional statements as well as some convenient built-in types, allowing us to substantially reduce model size and complexity. Figure 2 and Figure 3 showcase a particularly good example of this – in Figure 2, we take full advantage of K's syntax features and can express the full optimization problem in a fairly straightforward way. In Figure 3, we express only the interpolation constraint (the last constraint in **Error! Reference source not found.**) while restricting ourselves to just linear constraints, which is what the GNU Linear Programming Kit supports [6]. Expressing just the interpolation constraint using only linear constraints takes more code than expressing the entire model while making full use of K's features.

The advantages of this framework go beyond just convenience and speed. In Scenarios 3 and 4, we have constraints on the value for additional Δv. However, the value for additional Δv depends on an interpolated value that also circularly depends on the additional Δv itself. We could not find a straightforward way to converge on a satisfying solution by hand. The K/BAE framework was able to successfully find solutions for these scenarios.

One shortcoming of the current K/BAE framework is the inability to generate expressions as the solution for an optimization problem. For instance, in the 8-to-6 case, we know that the time required for 6 thrusters to operate is a function of the time accomplished on 8 thrusters, but the K/BAE framework can only solve for point solutions, like the minimum and maximum time on 6 thrusters. When adding another variable for delaying the start of the burn to investigate Scenarios 6 – 9, the expressions for the time bounds become much more complicated, and we decided that it would be too difficult to use the BAE to obtain the expressions. such that we could not simply transform the bound values into expressions. Continuing with the BAE would require software development for which did not plan.

```
//MINIMIZE
var y0 : Int
var y12 : Int
var y24 : Int
var y72 : Int

req 99999 * y0 - delayT >= 0
req 99999 * y12 - delayT >= - 12
req 99999 * y24 - delayT >= - 24
req 99999 * y72 - delayT >= - 72

//MAXIMIZE
var y0_12 : Int
req y0_12 - y0 <= 0
req y0_12 + y12 <= 1

var y12_24 : Int
req y12_24 - y12 <= 0
req y12_24 + y24 <= 1

var y24_72 : Int
req y24_72 - y24 <= 0
req y24_72 + y72 <= 1

//MINIMIZE
var slope : Real
var z0_12 : Int
req 99999 * z0_12 - slope + 21.2767 * x <= 0
req 99999 * z0_12 + slope - 21.2767 * x <= 0

var z12_24 : Int
req 99999 * z12_24 - slope - 24.4292 * x <= 37.83
req 99999 * z12_24 + slope + 24.4292 * x <= -37.83

var z24_72 : Int
req 99999 * z24_72 - slope + 10.2944 * x  <= -301.405
req 99999 * z24_72 + slope - 10.2944 * x <= 301.405

var x_notin_0_12_or_interp : Int
2 * x_notin_0_12_or_interp + y0_12 - z0_12 <= 2
2 * x_notin_0_12_or_interp + y0_12 - z0_12 >= 1

var x_notin_12_24_or_interp : Int
2 * x_notin_12_24_or_interp + y12_24 - z12_24 <= 2
2 * x_notin_12_24_or_interp + y12_24 - z12_24 >= 1

var x_notin_24_72_or_interp : Int
2 * x_notin_24_72_or_interp + y24_72 - z24_72 <= 2
2 * x_notin_24_72_or_interp + y24_72 - z24_72 >= 1

req additionalDeltaV - slope * (1.0 - completed) = 0.0

// set bounds for binaries used throughout model
req y0 >= 0
req y0 <= 1
req y12 >= 0
req y12 <= 1
req y24 >= 0
req y24 <= 1
req y72 >= 0
req y72 <= 1

req y0_12 >= 0
req y0_12 <= 1
req y12_24 >= 0
req y12_24 <= 1
req y24_72 >= 0
req y24_72 <= 1

req z0_12 >= 0
req z0_12 <= 1
req z12_24 >= 0
req z12_24 <= 1
req z24_72 >= 0
req z24_72 <= 1

req x_notin_0_12_or_interp >= 0
req x_notin_0_12_or_interp <= 1
req x_notin_12_24_or_interp >= 0
req x_notin_12_24_or_interp <= 1
req x_notin_24_72_or_interp >= 0
req x_notin_24_72_or_interp <= 1
```

Figure 3: Linear interpolation in K using only linear constraints. Compare this to the previous solution in Figure 2 that took full advantage of K's features and achieved this in just one constraint.

Therefore, in order to generate these expressions, we modeled the problem in Mathematica, using the Reduce utility to solve for desired variables. This approach required a preprocessing step which estimated how to sub-divide scenarios such that the resulting systems of equations were simple enough for Reduce to solve in a feasible time frame. Recombining these smaller solutions yields a numerically correct expression, at the cost of being too large and complex to yield insight into the problem. Furthermore, initial attempts to automatically simplify the expression proved too computationally expensive to be viable.

Mathematica is able to capture the K model given in a straightforward way. We did not choose to model in Mathematica originally since we already had a larger model of the spacecraft in K and thought that the expressions for the time bounds would be simple, as they are in Scenarios 1 – 4. Moreover, K/BAE has language constructs for activities and state/resource variables that help specify behavior at a higher level.

6. SUMMARY AND CONCLUSION

In our attempt to integrate more general behavior modeling with existing analytic models supporting probabilistic risk analyses, we encountered difficult problems in accounting for infinite fault cases in a closed-form calculation. Part of the problem is determining time bounds on when interdependent faults can occur in different scenarios, which feed into nested integrals for calculating reliability.

Using the declarative K language based on scenario and behavior ontologies with the BAE solver, we are able to solve 168 optimization problems to obtain nested time integral bounds for most of the scenarios of interest. We validated our results against those that we could solve by hand.

Taking advantage of K's features gave us simpler and more expressive models than those that other optimization toolkits would take. We were also able to plug into existing documentation tools making it possible to use this framework at a high level. A more detailed model of the spacecraft in K is used to determine undervoltage conditions.

One main limitation of the framework is that it solves for values, but we need to solve for expressions. The bounds on the amount of time spent on six thrusters actually depends on the amount of time spent on eight thrusters, so for the integrals, we would want to use these dependent expressions for the limits rather than just absolute values (see Appendix A). For more complicated cases (Scenarios 6 – 9), our approach was insufficient, but Mathematica was able to generate those formulas, although they are very large and difficult to manage.

APPENDIX

A. HAND CALCULATIONS

We initially hand calculated solutions to Scenarios 1 and 2 and used these results for validation (Tables 3 and 4, respectively).

Table 3: Hand calculated time bound values for Scenario 1.

Case	Δt	Δt_8	Δt_6	Δt_4
8 thrusters throughout	6.5	6.5	0	0
Switch from 8 to 6 thrusters	$6.5 < \Delta t \leq 7.5$	$3.5 \leq \Delta t_8 < 6.5$	$\frac{4}{3}(6.5 - \Delta t_8)$	0
Switch from 8 to 6 to 4 thrusters	$6.5 < \Delta t \leq 7.5$	$3.5 < \Delta t_8 < 6.5$	if $\Delta t_8 < 5.5$ *then* $11 - 2\Delta t_8 \leq \Delta t_6 < \frac{4}{3}(6.5 - \Delta t_8)$ if $\Delta t_8 \geq 5.5$ *then* $0 < \Delta t_6 < \frac{4}{3}(6.5 - \Delta t_8)$	$2\left(6.5 - \Delta t_8 - \frac{3}{4}\Delta t_6\right)$
Switch from 8 to 4 thrusters	$6.5 < \Delta t \leq 7.5$	$5.5 \leq \Delta t_8 < 6.5$	0	$2(6.5 - \Delta t_8)$
6 thrusters throughout	Not possible			
Switch from 6 to 4 thrusters	Not possible			
4 thrusters throughout	Not possible			

Table 4: Hand calculated time bound values for Scenario 2.

Case	Δt	Δt_8	Δt_6	Δt_4
8 thrusters throughout	Not possible			
Switch from 8 to 6 thrusters	$7.5 < \Delta t < \frac{4}{3} \cdot 6.5$	$0.62 < \Delta t_8 < 3.5$	$\frac{4}{3}(6.5 - \Delta t_8)$	0
Switch from 8 to 6 to 4 thrusters	$7.5 < \Delta t < 2 \cdot 6.5$	$0.62 < \Delta t_8 < 5.5$	if $\Delta t_8 < 4.54$ *then* $4.54 \cdot 2 - 2\Delta t_8 \leq \Delta t_6 < \frac{4}{3}(6.5 - \Delta t_8)$ if $\Delta t_8 \geq 4.54$ *then* $0 < \Delta t_6 < \frac{4}{3}(6.5 - \Delta t_8)$	$2\left(6.5 - \Delta t_8 - \frac{3}{4}\Delta t_6\right)$
Switch from 8 to 4 thrusters	$7.5 < \Delta t < 2 \cdot 6.5$	$4.54 < \Delta t_8 < 5.5$	0	$2(6.5 - \Delta t_8)$
6 thrusters throughout	Not possible			
Switch from 6 to 4 thrusters	Not possible			
4 thrusters throughout	Not possible			

ACKNOWLEDGEMENTS

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

We would like to thank Cameron Burnett for his major contribution to behavior modeling.

REFERENCES

[1] M. DiNicola, K. McCoy, C. Everline, K. Reinholtz, and E. Post, "A mathematical model for assessing the probability of contaminating Europa," in 2018 IEEE Aerospace Conference, 2018, pp. 1–20.

[2] G. F. Dubos, D. P. Coren, A. Kerzhner, S. H. Chung, and J. Castet, "Modeling of the flight system design in the early formulation of the Europa Project," in 2016 IEEE Aerospace Conference, 2016, pp. 1–14.

[3] C. J. Everline, "Bayesian Approach to Quantifying Epistemic Uncertainty in a Processor Availability model," Journal of Spacecraft and Rockets, vol. 49, no. 6, pp. 1019–1031, 2012.

[4] C. J. Everline and T. Paulos, "Comparison of techniques for modeling accident progression in dynamic aerospace applications with and without repair," Reliability Engineering & System Safety, vol. 91, no. 3, pp. 370–377, Mar. 2006.

[5] C. J. Everline et al., "Estimating the reliability of electronic parts in high radiation fields," May 2008.

[6] GNU, "GNU Linear Programming Kit," Free Software Foundation, Inc., 2018

[7] K. Havelund, R. Kumar, C. Delp, and B. Clement, "K: A wide spectrum language for modeling, programming and analysis," in 2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), 2016, pp. 111–122.

[8] J. Kaderka, M. Rozek, J. Arballo, D. Wagner, and M. Ingham, "The Behavior, Constraint, and Scenario (BeCoS) Tool: A Web-Based Software Application for Modeling Behaviors and Scenarios," in 2018 AIAA Aerospace Sciences Meeting, Kissimmee, Florida, 2018.

[9] K. McCoy, B. Nairouz, B. Bradley, L. Jones-Wilson, and S. Susca, "Assessing the science robustness of the Europa clipper mission: Science sensitivity model," in 2018 IEEE Aerospace Conference, 2018, pp. 1–13.

[10] B. V. Oaida, K. Lewis, E. Ferguson, J. Day, and K. McCoy, "A statistical approach to payload energy management for NASA's Europa Clipper mission," in 2018 IEEE Aerospace Conference, 2018, pp. 1–12.

[11] T. Paulos, "A Discussion of Failure Mode Modeling of Complex Components and Overall Component Reliability," PSAM 2016, p. 8, 2016.

[12] S. Schreiner, M. L. Rozek, A. Kurum, C. J. Everline, M. D. Ingham, and J. Nunes, "Towards a methodology and tooling for Model-Based Probabilistic Risk Assessment (PRA)," in AIAA SPACE 2016, American Institute of Aeronautics and Astronautics, 2016.

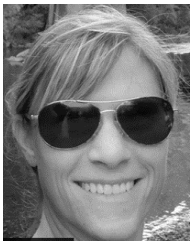
BIOGRAPHY

Anthony Zheng *is currently pursuing a B.S.E. in Computer Science Engineering at the University of Michigan College of Engineering in Ann Arbor. He spent his summer in 2018 working as an intern at the NASA Jet Propulsion Laboratory.*

Brad Clement *is a senior member of the technical staff at the Jet Propulsion Laboratory in Pasadena, CA. He received a bachelor degree in computer engineering from the Georgia Institute of Technology and M.S. and Ph.D. degrees in computer science and engineering from the University of Michigan, Ann Arbor. His interests include artificial intelligence, planning, scheduling, multiagent coordination, uncertainty in AI, robotics, distributed and real-time systems, and control systems.*

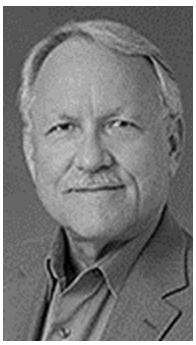
David Legg *is currently pursuing a B.S. in Mathematics and Computer Science at University of California, Irvine. He spent the summers of 2017 and 2018 as an intern at the NASA Jet Propulsion Laboratory, where he supported ongoing efforts to integrate engineering models and analysis tools.*

Mitch Ingham *is a senior software system engineer in the Flight Software Systems Engineering and Architecture Group at the Jet Propulsion Laboratory. His research interests include model-based methods for systems and software engineering, software architectures, and spacecraft autonomy. He earned his Sc.D. and S.M degrees from MIT in Aeronautics and Astronautics, and a B.Eng. in Honours Mechanical Engineering from McGill University in Montreal, Canada.*



Kelli McCoy *began her career at NASA Kennedy Space Center in 2004 as an Industrial Engineer in the Launch Services Program, following her graduation from Georgia Tech with a M.S in Industrial and Systems Engineering. She went on to obtain a M.S. in Applied Math and Statistics at Georgetown University, and subsequently developed probability models to estimate cost and schedule during her*

tenure with the Office of Evaluation at NASA Headquarters. Now at Jet Propulsion Laboratory, she has found applicability for math and probability models in an engineering environment and is currently developing that skillset as the Lead of the Europa Project Systems Engineering Analysis Team.



***Chet Everline**'s expertise is in probabilistic risk assessment to support system trades and demonstrate compliance with quantitative requirements (such as those for planetary protection or orbital debris). At JPL he has supported the MER, Juno, SMAP, ARRM, Clipper, and Mars 2020 missions. He contributed to the Hubble Space Telescope system reliability review, was a member of the Space Launch System PRA independent peer review teams, and served on the peer review panels for STEP 4 system safety certification.*

Prior to joining JPL he had a key role in formulating the approach selected for disposal of US chemical munitions and the Department of Energy's recommendations regarding demilitarizing the US nuclear arsenal. Chet is currently the JPL point of contact for probabilistic risk assessment.